Footprints and Free Space from a Single Color Image

Jamie Watson[1] Michael Firman[1] Aron Monszpart[1] Gabriel J. Brostow[1,2]
[1]Niantic [2]UCL
www.github.com/nianticlabs/footprints

Abstract

Understanding the shape of a scene from a single color image is a formidable computer vision task. However, most methods aim to predict the geometry of surfaces that are visible to the camera, which is of limited use when planning paths for robots or augmented reality agents. Such agents can only move when grounded on a traversable surface*, which we define as the set of classes which humans can also walk over, such as grass, footpaths and pavement. Models which predict beyond the line of sight often parameterize the scene with voxels or meshes, which can be expensive to use in machine learning frameworks.*

We introduce a model to predict the geometry of both visible and occluded traversable surfaces, given a single RGB image as input. We learn from stereo video sequences, using camera poses, per-frame depth and semantic segmentation to form training data, which is used to supervise an image-to-image network. We train models from the KITTI driving dataset, the indoor Matterport dataset, and from our own casually captured stereo footage. We find that a surprisingly low bar for spatial coverage of training scenes is required. We validate our algorithm against a range of strong baselines, and include an assessment of our predictions for a path-planning task.

1. Introduction

Computerized agents, for example a street cleaning robot or an augmented reality character, need to know how to explore both the *visible* and the *hidden, unseen* world. For AR agents, all paths must be planned and executed without camera egomotion, so no new areas of the real scene are revealed as the character moves. This makes typical approaches for path planning in unknown [65, 74] and dynamic environments less effective.

We introduce Footprints, a model for estimating both the visible and hidden traversable geometry given just a single color image (Figure 1). This enables an agent to know where they can walk or roll, beyond the immediately visible surfaces. Importantly, we model not just the surfaces' overall shapes and geometry [21, 22], but also where moving and static objects in the scene preclude walking. We refer to these occupied regions of otherwise traversable surfaces

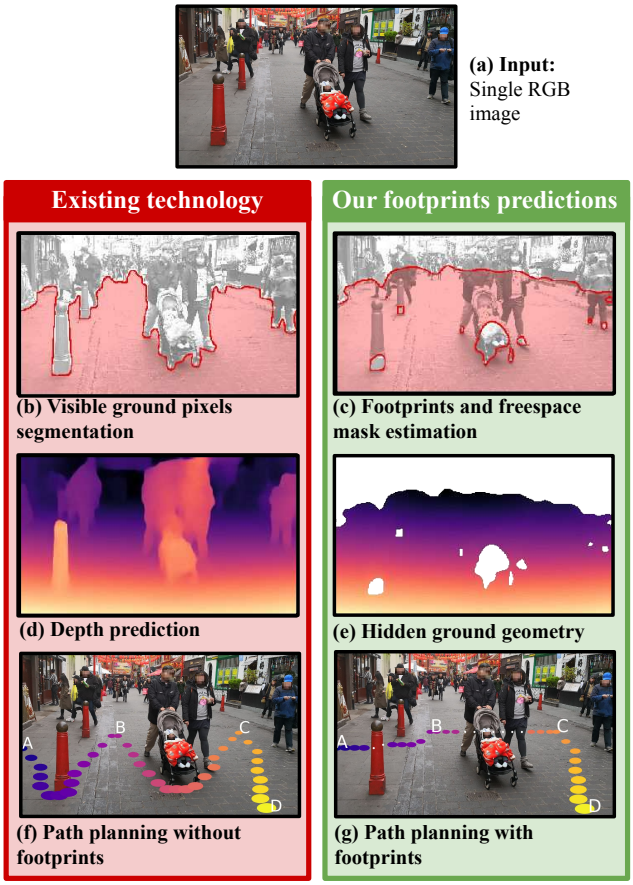


Figure 1. **Footprints overview:** Given a single color image (a), existing methods can estimate the segmentation of which *visible* pixels can be traversed by a human or virtual character (b) and the depth to each pixel (d). We introduce **Footprints**, a method to estimate the extent (c) and geometry (e) that includes *hidden* walkable surfaces. Our predictions can be used, for example, to plan paths through the world. Here we plan a path from $A \rightarrow B \rightarrow C \rightarrow D$ using the ground predictions, with the A* algorithm [26]. The baseline path (f) takes an unrealistic route sticking only to *visible ground surface*. Our hidden geometry predictions enable a realistic path behind objects to be found (g).

as object *footprints*.

Previous approaches rely on bounding box estimates [27, 36, 57], which are limited to cuboid object predictions. Other approaches to estimating missing geometry have required *complete*, *static* training environments, which have either been small in scale [10] or synthetic [6, 63]. Surprisingly, our method can create plausible predictions of hidden surfaces given only partial views of real moving scenes at training time. We make three contributions:

1. We introduce a lightweight representation for hidden geometry estimation from a single color image, with a method to learn this from video depth data.
2. We present an algorithm to learn from videos with moving objects and incomplete observations of the scene, through masking of moving objects, a prior on missing data, and use of depth to give additional information.
3. We have produced human-annotated hidden surface labels for all 697 images in the KITTI test set [16]. These are available to download from the project website. We also introduce evaluation methods for this task.

2. Related Work

Our method is related to prior work in robotics, path planning, and geometry estimation and reconstruction.

2.1. Occupancy maps and path planning

If multiple camera views of a scene are available, camera poses can be found and a 3D model of a static scene can be reconstructed [45]. The addition of a segmentation algorithm enables the floor surface geometry to be be found [1, 41]. In our work, we make floor geometry predictions given just a single image as input. Other multi-view approaches include occupancy maps in 2D [58] and 3D [46, 67, 75], where new observations are fused into a single map.

The planning of paths of virtual characters or robots in environments with known geometry is a well-studied problem [5, 18, 33, 54, 66]. Our prediction of walkable surfaces beyond the line of sight shares concepts with works which allow for path planning in environments where *not all geometry can be observed* [65, 74]. Gupta *et al*. [24] learn to plan paths with a walkable geometry belief map similar to our world model, while [34] learn potential navigable routes for a robot from watching video. Rather than directly planning paths, though, in our work we directly learn and predict geometry, which is useful for path planning and more.

2.2. Predicting geometry you *can* see

A well-studied task for geometry estimation is the prediction of a depth map given a single color image as input. The best results here come from supervised learning, e.g. [9, 14]. Acquiring supervised data for geometry estimation is hard, however, so a popular approach is self-supervised learning, where training data can be monocular [20, 52, 79] or stereo [15, 19, 49, 76] images. Depths are learned by minimising a reprojection loss between a target image and a warped source view. Like these works, we also learn from arbitrary videos to predict geometry, but our geometry predictions extend *beyond the line of sight of the camera*.

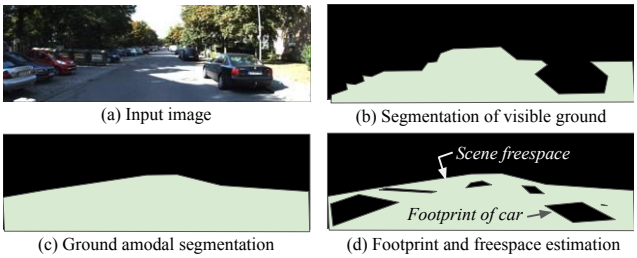
2.3. Predicting geometry you *can't* see

We fall into the category of works which predict geometry for parts of the scene which are not visible in the input view. For example, [48, 64] perform view extrapolation, where semantics and geometry outside the camera frustum are predicted. In contrast, we make predictions for geometry which is *inside* the camera frustum, but which is *occluded* behind objects in the scene.

Geometry completion Predicting the occupancy of unobserved voxels from a single view is one popular representation for hidden geometry prediction [6, 10, 63]. Training data for dense scene completion is difficult to acquire, though, often making synthetic data necessary [6, 63]. Further, voxels can be slow to process and computation hard to scale for geometry prediction, making their use in real-time or on mobile platforms difficult. Meshes are a more lightweight representation [61] but incorporating meshes in a learning framework is still an active research topic; a typical approach is to go via an intermediary voxel representation, e.g. [17]. A complementary source of information is physical stability as a cue to complete scenes [59].

Layered completion Recent works have taken a lightweight approach to predicting hidden scene structure by decomposing the visible image into layers of color and depth behind the immediately visible scene [8, 40, 60, 68]. Similarly, amodal segmentation [12, 51, 80] aims to predict overlapping semantic instance masks which extend beyond the line of sight. However, amodal segmentation doesn't label the contact points necessary to know the location of objects. Amodal segmentation would label a 'traversable surface' as continuous under a car or person.

Floor map prediction Similar to amodal segmentation are approaches that predict the floor map from a single color image, for example [55, 71]. Similarly [21, 22] complete support surfaces in outdoor and indoor scenes respectively. The aim of these approaches is to predict support surfaces as if all objects were absent (Figure 2(c)), akin to amodal segmentation, while we aim to predict the walkable floor surface taking obstacles into account (Figure 2(d)). The Manhattan layout assumption can be useful to help infer the ground surface in indoor scenes (e.g. [27, 35, 36, 57]), however, is less applicable outdoors. Our task is motivated by prior work [72], though our approach is novel.



(a) Input image (b) Segmentation of visible ground (c) Ground amodal segmentation (d) Footprint and freespace estimation

Figure 2. For an input image, segmentation (b) only captures traversable surface visible from this viewpoint, while amodal segmentation (c) fails to delineate which parts of the ground cannot be traversed due to the presence of objects. Our goal (d) is to capture the free, traversable space in the scene and the footprints of objects which preclude motion.

Detection approaches One method to estimate the full extent of partially observed objects is via 3D detection, for example 3D bounding boxes [32, 37, 39, 53, 62]. Generic object bounding box detectors have been used to estimate indoor free space [28, 36, 57]. Bounding boxes only give convex footprints for 'things' in the image, so aren't suitable for the geometry of 'stuff' [2] such as walls, piles of items, or shrubbery. To the best of our knowledge, object detection has not been effectively combined with amodal segmentation to give traversable surfaces. We compare to recent object detection baselines and show that our approach is better suited to our task (Section 5). Another detection approach is to fit 3D human models to help estimate the hidden layout [13, 42], while our aim also has similarities to [23], who aim to recover the places in a scene a human can stand, sit and reach. Such methods often operate with a static scene assumption and work best when the whole scene has been "explored" by the humans.

In comparison to these related works, we predict the hidden and visible traversable surfaces from a single image, taking all obstacles (whether 'things' or 'stuff') into account.

3. Our Footprints world model

Our goal is to predict both the visible and hidden traversable surface for a single color image I_t. A surface is defined as traversable if it is visually identifiable as one of a predefined set of semantic classes, listed in our supplementary material. The **visible traversable surface** can be represented with two single-channel maps:

1. A visible ground segmentation mask S. Each $s_j \in S$ is 1 if the surface seen at pixel j is from a traversable class, and 0 otherwise. S can be estimated with e.g. [25, 77].
2. A visible depth map D giving the distance from the camera to each visible pixel in the scene, e.g. [19].

Together, $\{D, S\}$ model the extent and geometry of all the *visible* ground which can be traversed – Figure 2(b). However, to know about how an agent could move through areas of the scene beyond the line of sight, we also need to model geometric information about ground surfaces which are occluded by objects. To this end, our representation also incorporates two channels which model the **hidden traversable surface**:

3. A *hidden* ground segmentation mask S^*, which represents the extent of the entire traversable floor surface inside the camera frustum, including occluded parts. Each pixel $s^*_j \in S^*$ is 1 if the camera ray associated with pixel j intersects with a *walkable* surface at any point (even behind objects visible in this view) and 0 otherwise. This can also be seen as a top-down floor map reprojected into the camera view [24].
4. A depth map D^* which gives the geometry of the hidden ground surface. Each $d^*_j \in D^*$ contains the depth from the camera to the (visible or hidden) ground for pixel j. If the camera ray at pixel j doesn't intersect any traversable surface (*i.e.* $s^*_j = 0$), then d^*_j is 0.

Our four-channel representation $\{S, D, S^*, D^*\}$ is a rich world model which enables many tasks in robotics and augmented reality, while being lightweight and able to be predicted by our standard image-to-image network.

How does our model relate to ground segmentation? A semantic segmentation algorithm also gives us the pixels which an agent could walk on, but only those which are visible by the camera (*i.e.* S). Our model also represents the location of walkable ground surfaces which are not visible to the camera.

Why can't we just fit a plane? Assuming a planar floor surface, fitting a plane to the visible ground would give an estimate of the *geometry* of the walkable surface. However, this planar model does not give the *extents* of the walkable surface, meaning an agent traversing the scene would walk into objects.

Why not use a voxel model? Our image-space predictions are lightweight and memory efficient, and furthermore output is pixel-wise aligned with the input space. Given that our main focus is on where we can walk, our representation is the minimal necessary representation.

Why not make the predictions in top-down space? We could represent the world in top-down view instead of in reprojected camera space. While this would allow us to model the world outside the camera frustum, we would add complexity, with more complicated training and reliance on good test-time camera-pose estimation.

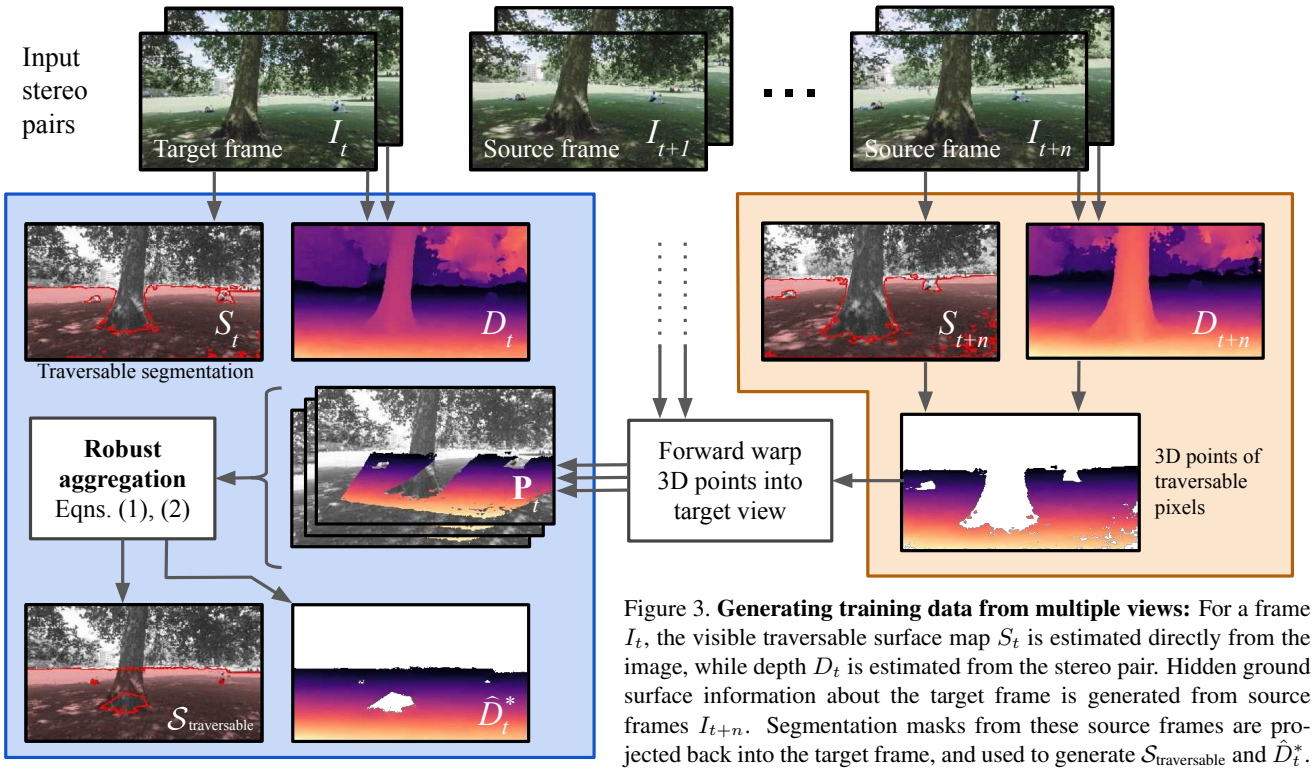


Figure 3. **Generating training data from multiple views:** For a frame I_t, the visible traversable surface map S_t is estimated directly from the image, while depth D_t is estimated from the stereo pair. Hidden ground surface information about the target frame is generated from source frames I_{t+n}. Segmentation masks from these source frames are projected back into the target frame, and used to generate $\mathcal{S}_{\text{traversable}}$ and $\hat{D}_t^*$.

4. Learning to predict Footprints

It is possible to estimate $\{S, D\}$ using off-the-shelf prediction models, e.g. [31]. However, training a model to estimate $\{S^*, D^*\}$ requires additional sources of information. Human labeling is expensive and difficult to do at scale as we are asking an annotator to label occluded parts of a scene. Instead, we exploit two readily available sources of information: freely captured video and depth data. We use these to divide pixels from each training image into three disjoint sets. $\mathcal{S}_{\text{traversable}}$ contains indices of pixels which are deemed to be traversable; $\mathcal{S}_{\text{untraversable}}$ the indices of pixels which we are confident cannot be traversed, and $\mathcal{S}_{\text{unknown}}$ the indices of pixels which we have no information about. These *unknown* predictions come about by our use of freely captured video for training; some areas of the scene have never been observed, and we have no information about whether these regions are traversable or not.

4.1. Learning $\mathcal{S}_{\text{traversable}}$ from video data

Freely captured video is easy to obtain and gives us the ability to generate training data for geometry behind visible objects. We use other frames in the video to provide information about what the geometry and shape of the walkable surface is by projecting observations from each frame back into the target camera.

We use off-the-shelf tools to estimate camera intrinsics and depth maps for each frame and relative camera poses between source frames I_{t+i} and the target frame I_t. We then forward warp [56, 70] the depth values of traversable pixels from the source frame into the target frame. This results in a sparse depth map $P_{t+i \to t}$, representing the geometry and extents of the traversable ground visible in frame I_{t+i} rendered from the viewpoint of I_t. We repeat this forward-warping for N nearby frames, obtaining the set $\mathbf{P}_t = \{P_{t+i \to t}\}_{i=1}^{N}$.

Due to inaccuracies in floor segmentation, depth maps, and camera poses, many of the reprojected floor map images $P_{t+i \to t}$ will have errors. We therefore perform a *robust* aggregation of the multiple noisy segmentation and depth maps to form a single training image. Our traversable labelset $\mathcal{S}_{\text{traversable}}$ is formed from pixels for which at least k reprojected depth maps contain a nonzero value, *i.e.*

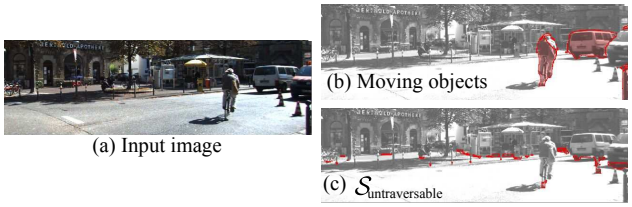
$$\mathcal{S}_{\text{traversable}} = \left\{ j \in \mathcal{J} \mid \left(\sum_{P \in \mathbf{P}_t} [p_j > 0] \right) > k \right\}, \quad (1)$$

where [] is the Iverson bracket, $\mathcal{J}$ is the set of all pixel indices in this image and p_j is the jth pixel in P. See Figure 3 for an overview.

We subsequently obtain our ground depth map $\hat{D}^*$ by taking the median depth value (ignoring zeros) associated with each pixel j if and only if there is a valid depth value at this location:

$$\hat{D}^* = \text{median}\big(\{P \in \mathbf{P}_t \mid P > 0\}\big) \quad (2)$$

We supervise our prediction $\hat{D}^*$ with a $\log L_1$ loss [29].



(a) Input image

Figure 4. **Moving object and depth masking:** For a training image (a), our moving object mask (b) identifies pixels associated with moving objects. The set of pixels $\mathcal{S}_{\text{untraversable}}$ (c) uses the training depth image to capture the footprints of small and thin objects.

4.2. Depth masking to find $\mathcal{S}_{\text{untraversable}}$

While $\mathbf{P}_t$ is constructed from depth images of multiple source images, models trained on $\mathbf{P}_t$ alone typically incorrectly estimate object footprint boundaries, often entirely missing the footprints of thin objects such as poles and pedestrians. Such mistakes are due to inaccuracies in camera pose tracking, traversable segmentation masks and visible depth maps, resulting in sometimes poor reprojections into the target frame that are not excluded by our robust aggregation method. To tackle this problem, we exploit depth data from the target image I_t to estimate $\mathcal{S}_{\text{untraversable}}$, the set of pixels in the image which are *definitely not* traversable. Subsequently, we redefine $\mathcal{S}_{\text{traversable}}$ to not include pixels in $\mathcal{S}_{\text{untraversable}}$.

To find $\mathcal{S}_{\text{untraversable}}$, we first project all points in the depth map D_t from camera space into world space. Next, we fit a plane to those points which are classified as visible ground in our segmentation mask S_t using RANSAC [11]. We then move each point in the world along the normal vector of the plane such that they now lie on the plane, and 'splat' in a small grid around the resulting position. After reprojecting these points back into camera space, we apply a filtering step (see supplementary material for details) to remove erroneous regions, and obtain the set of pixels $\mathcal{S}_{\text{untraversable}}$. An example is shown in Figure 4(c).

4.3. Masking moving objects at training time

The computation of $\mathcal{S}_{\text{traversable}}$ and $\hat{D}^*$ utilizes multiple frames and makes the significant and unrealistic assumption that our training data comes from a static world, when in fact many objects will undergo significant motion between frames I_t and I_{t+i}. To combat this, we identify and remove pixels from our training loss which are associated with moving objects. We could use semantic segmentation to remove non-static object classes, such as cars; however, this would prevent us learning about the hidden geometry of *any* cars, including parked ones. We could train on static scenes [38], but would be limited by the availability of existing general-purpose datasets. We instead observe that most classes of moving objects are static at least some of the time. For example, while it is hard to learn the geometry of a moving vehicle, we can learn the shape of parked cars and apply this knowledge to moving cars at test time. Similarly, footprints of humans can be learned by observing those which are relatively static in training.

We compute a per-pixel binary mask M, where $\mu_j \in M$ is zero for pixels depicting non-static objects. To compute M for frame t, we computed the *induced flow* [69, 81] from frame t to $t+1$, using D_t and camera motion. This estimated where pixels would have moved to assuming a static scene. We also separately estimate frame-to-frame optical flow. Pixels where the induced and optical flow differ are often pixels on moving objects; we set μ_j to 0 if the endpoints of the two flow maps differ by more than $\tau = 3$ pixels, and 1 otherwise. An example of M is shown in Figure 4(b).

4.4. Final training loss

Our training loss comprises four parts, one for each output channel $\{S^*, D^*, S, D\}$.

Hidden traversable surface loss $l_j^{s^*}$ —

$$l_j^{s^*} = \begin{cases} -\mu_j \log(\hat{s}_j^*) & \text{if } j \in \mathcal{S}_{\text{traversable}} \quad (3a) \\ -\log(1-\hat{s}_j^*) & \text{if } j \in \mathcal{S}_{\text{untraversable}} \quad (3b) \\ -\lambda \log(1-\hat{s}_j^*) & \text{otherwise,} \quad (3c) \end{cases}$$

where (3a) encourages pixels in $\mathcal{S}_{\text{traversable}}$ to be labelled $s_j^* = 1$; (3b) encourages pixels in $\mathcal{S}_{\text{untraversable}}$ to be labelled $s_j^* = 0$; and (3c) applies a prior $\lambda < 1$ to the remaining, unknown pixels which conservatively encourages them to be labeled as untraversable.

Visible traversable surface loss l_j^s — This is supervised using standard binary cross-entropy loss.

Observed depth loss l_j^d — For the channel predicting the depth map for visible pixels, we follow [29, 73] and supervise with $l_j^d = \log(|d_j - \hat{d}_j| + 1)$.

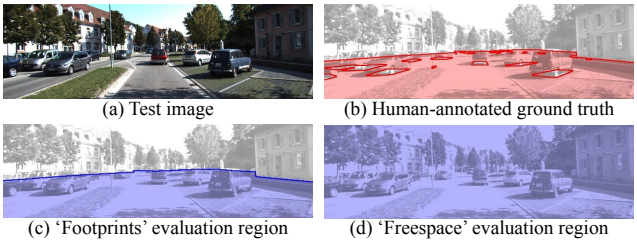
Hidden depth loss $l_j^{d^*}$ — Hidden depths are also supervised with the log L_1 loss, but we only apply the loss for pixels $\in \mathcal{S}_{\text{traversable}}$.

Our final loss is the sum of each subloss over all pixels:

$$L = \sum_j l_j^{s^*} + l_j^s + l_j^d + l_j^{d^*}. \quad (4)$$

4.5. Implementation details

To generate training signals for KITTI and our casually captured stereo data, camera extrinsics and intrinsics are estimated using ORB-SLAM2 [44], while depth maps are inferred from stereo pairs using [4]. Segmentation masks are estimated using a simple image-to-image network trained



(a) Test image (b) Human-annotated ground truth

(c) 'Footprints' evaluation region (d) 'Freespace' evaluation region

Figure 5. **Evaluation region:** Object footprints are evaluated on all pixels within the human-annotated ground polygon (a), while for freespace evaluation we use the whole image (b); see Sec 5.

using the ADE20K [78] and Cityscapes [7] datasets, and optical flow is estimated using [30, 47]. Our network architecture is based on [20], modified to predict four sigmoided output channels. We adjust our training resolution to approximately match the aspect ratio of the training images: 512×640 for the Matterport dataset, 192×640 for KITTI, and 256×448 for our own stereo data. For Matterport, camera intrinsics, relative locations, and depth maps are provided. Thus we need only estimate segmentation masks, and do so using the same pretrained network finetuned on a small subset of 5,000 labelled Matterport images. Except in some ablations, we set $\lambda = 0.25$.

5. Experiments

We validate our scene representation and the associated learning approach experimentally. We do this by:

a) Quantifying the accuracy of our predictions indoors and outdoors (**Matterport** and **KITTI**),
b) Illustrating their quality across different scenarios,
c) Ablating to gauge the benefits of different design decisions, and
d) Illustrating a use-case where Footprints are used for path planning (Sec 6).

We focus our evaluation here on hidden traversable surface estimation *i.e.*, S^*, and we evaluate S, D and D^* in the supplementary material.

Metrics: There are two aspects of S^* predictions which are of interest: (1) The ability to estimate the overall extents of the traversable freespace in the image, and (2) the ability to estimate the footprint base of objects in the scene which must be avoided. To capture this we introduce two evaluation settings. The first, *freespace evaluation*, addresses (1) by evaluating our thresholded prediction of S^* over all pixels in the image using the standard binary detection metrics of IoU and F1. The second is *footprints evaluation* addressing (2), where we focus on the evaluation of object footprints by evaluating only within the ground region. To evaluate all methods equivalently, we evaluate within the true ground segmentation (KITTI) and the convex hull of the true visible ground (Matterport) — see Figure 5.

	Freespace eval.		Footprint eval.	
	IoU	F1	IoU	F1
Convex hull	0.790	0.876	0.145	0.230
Bounding box	<u>0.794</u>	<u>0.879</u>	0.187	0.292
Nothing traversable ($S^* = 0$)	0.000	0.000	0.089	0.153
Everything traversable ($S^* = 1$)	0.344	0.506	0.000	0.000
Visible ground	0.770	0.860	<u>0.231</u>	<u>0.356</u>
Ours	**0.797**	**0.880**	**0.239**	**0.363**

Table 1. **Evaluating object footprint and freespace detection on the KITTI dataset:** Best methods in each category are **bolded**; second best <u>underlined</u>. Our method outperforms all baselines.

	Freespace eval.		Footprint eval.	
	IoU	F1	IoU	F1
Project down baseline	0.344	0.506	0.082	0.144
Ours w/o moving object masks	0.795	0.878	0.227	0.347
as above w/o eqn. (3b)	**0.797**	<u>0.879</u>	0.218	0.333
Ours w/o eqn. (3b)	0.793	0.877	0.225	0.343
Ours ($\lambda = 0$)	0.355	0.519	0.217	0.335
Ours ($\lambda = 0.5$)	0.787	0.873	0.232	0.355
Ours ($\lambda = 1.0$)	0.776	0.865	<u>0.234</u>	<u>0.356</u>
Ours	**0.797**	**0.880**	**0.239**	**0.363**

Table 2. **Ablating our method on the KITTI dataset:** Our ablations validate our approach; removing components of our method gives equivalent *freespace* scores, but are significantly worse at detecting object *footprints*.

Baselines: We compare against several baselines, to demonstrate the efficacy of our method across tasks:

Visible only — S^* is set as the visible ground mask S.

Convex hull — We estimate S^* as the convex hull of the visible ground mask S.

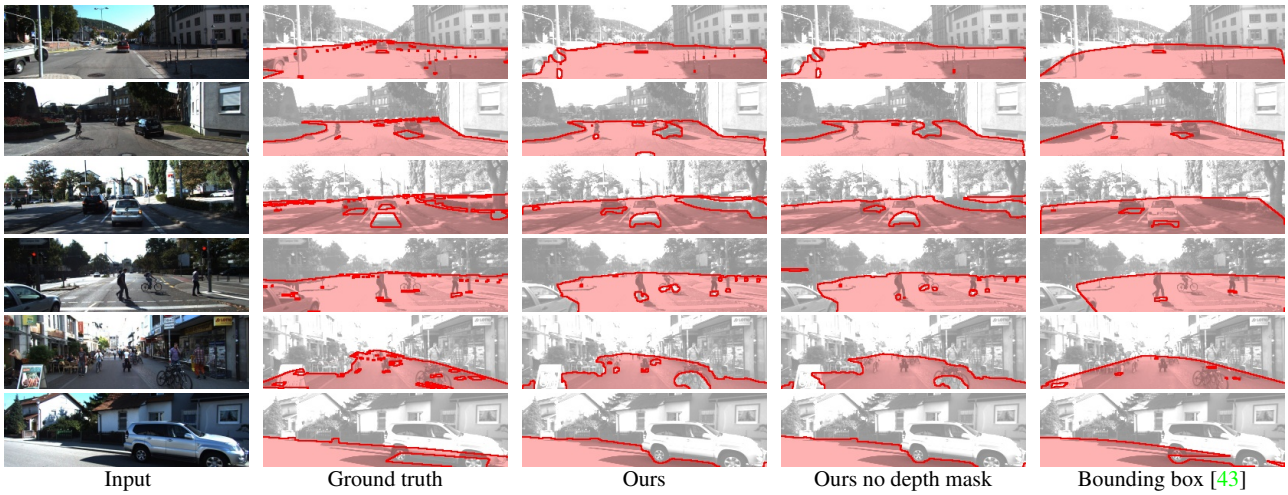
3D bounding boxes — Footprints of objects are estimated using 3D bounding box detectors [43] for outdoor scenes and [50] indoors; we evaluate both the 'ScanNet' and 'SunRGBD' models from [50]. Estimated object footprints are subtracted from the convex hull baseline for the final prediction. Unlike our method, [50] make predictions with access to the structured-light-inferred depth map at test time; we include their state-of-the-art results as an upper bound on what a bounding box method could achieve.

Voxel prediction — On indoor scenes, we use [63] to estimate the voxelized scene from a depth input. Voxels estimated as 'floor' are reprojected into the camera.

Project down — We train a model to estimate footprints using only depth images at training time, without our multi-frame reprojection. For this we train a binary classifier to predict if it expects each pixel to be a member of $\mathcal{S}_{\text{untraversable}}$ or not, and subtract these pixels from the convex hull.

5.1. Evaluation on the KITTI benchmark

We first train and evaluate on the well-established **KITTI** benchmark [16] using the Eigen split [9]. To evaluate quantitatively, we generate human annotations for the entire test set. Labelers were instructed to draw a polygon

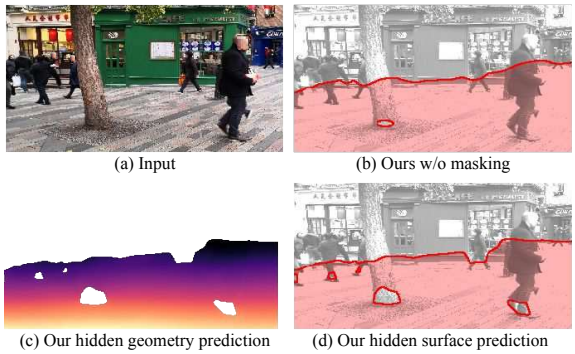


Input | Ground truth | Ours | Ours no depth mask | Bounding box [43]

Figure 6. **KITTI results:** Each row shows an input image and the predicted S^* mask from our model, our model without depth masking and the strongest baseline. We find the footprints of a wider variety of objects than the baseline, and are better at capturing the overall shape of the traversable space. The 1st and 4th rows show the benefits of depth masking for thin objects. The final row shows a failure, where we fail to predict walkable space behind a car; our facing-forward training videos means our network rarely sees behind some objects.

	Freespace eval.		Footprint eval.	
	IoU	F1	IoU	F1
Nothing traversable ($S^* = 0$)	0.000	0.000	0.186	0.291
Everything traversable ($S^* = 1$)	0.480	0.611	0.000	0.000
Convex hull	0.454	0.562	0.289	0.421
Bounding box† [50] (Scannet)	0.450	0.557	0.333	0.469
Bounding box† [50] (Sun RGBD)	0.451	0.559	0.315	0.450
Voxel SSCNet† [63]	0.492	0.615	0.087	0.136
Voxel SSCNet+†	0.418	0.547	0.107	0.173
Visible ground ($S^* = S$)	0.505	0.628	0.404	0.542
Ours	**0.652**	**0.767**	**0.426**	**0.557**
Ours ($\lambda = 0.5$)	**0.663**	**0.776**	**0.452**	**0.585**

Table 3. **Evaluation of S^* on Matterport [3]:** We are the best method in both freespace and footprint evaluation across all metrics. The final row shows an ablation which outperforms *ours*, suggesting that a careful choice of hyperparameters could further improve performance. Methods marked † have access to structured-light depth data at test time. Voxel SSCNet(+)'s geometry estimation failed on 178 scenes; we ignore these when averaging.



(a) Input | (b) Ours w/o masking | (c) Our hidden geometry prediction | (d) Our hidden surface prediction

Figure 7. **Stereo capture results:** Prediction of our model trained on handheld stereo footage for an image taken using a mobile phone. Here we see the qualitative impact of using our full method (c), (d) vs. with neither depth masking nor moving object masking (b). We are better able to capture the footprints of pedestrians.

bounding the hidden and visible walkable surface, and to separately label the footprint of each occluding object in the scene. Due to the nature of our task, labelers had to estimate the hidden extents of many objects, which seems like an error-prone task. However, this follows work in amodal labeling where consistency between labelings was found to be reasonably high [51]. These annotations are available from the project website.

We present quantitative results of our method alongside baselines in Table 1. Here we demonstrate the superior performance of our method in both freespace and footprint evaluation. Qualitative results can be seen in Figure 6. We see that *Ours* finds the footprints of a wider variety of objects than *Bounding Box* as we are not limited to predefined classes. We also better capture the overall shape of the traversable ground. Additionally, we ablate our method in Table 2, showing that our full method helps to improve results.

5.2. Indoor evaluation

We use the **Matterport** dataset [3] for training and evaluation on indoor scenes. Here, camera poses and structured-light depth maps are provided, and the ground truth floor masks and geometry are rendered from the dataset's semantically annotated mesh representation. We only train and evaluate on images from the forward- and downward-facing cameras on their rig, leaving us with 49,286 training images. We evaluate on the first 500 images from the test set. Results are shown in Figure 8 and Tables 3 and 4, where we again outperform all baselines. SSCNet [63] performs poorly as this method was mainly trained on synthetic data, where the footprints of objects are not separately delineated from the ground plane. We therefore create a reworking of their method, SSCNet+. Here, the voxel predictions of

	a_1	RMSE	Abs. rel.	Sq. rel.
SSCNet† [63]	0.069	6.689	1.434	14.667
RANSAC plane	0.359	1.713	0.307	0.865
Ours	**0.577**	**1.101**	**0.206**	**0.292**
RANSAC (oracle*)	0.351	1.693	0.306	0.821

Table 4. **Matterport hidden depth (D^*) evaluation:** Our method outperforms the baselines, even the artificially boosted method in the final row which has access to ground truth visible ground segmentation and ground truth depths.

	Preprocessing (s)	Inference (s)
Voxels (SSCNet) [63]	43	66
Bounding box [50]	-	0.417
Bounding box [43]	-	0.520
Ours	-	**0.074**

Table 5. **Single image inference speed comparison:** Our image-to-image network is significantly faster than alternative off-the-shelf 3D geometry estimation methods.

	Failed paths	Collisions
SSCNet [63]	0.643	0.207
Convex hull	0.608	0.180
Bounding box [50] (Scannet)	0.569	0.157
Bounding box [50] (Sun RGBD)	0.575	0.162
Predicted visible ground	0.512	0.126
Nothing traversable ($S^* = 0$)	0.616	0.198
Ours	**0.498**	**0.109**
Ground truth	0.255	0.040

Table 6. **Path planning evaluation on the Matterport dataset:** 'Collisions' averages the total fraction of each path spent in space marked as non-traversable by the ground truth, while a path is 'failed' if it leaves ground-truth traversable space at any single point. Lower scores are better in both columns.

chairs, beds, sofas, tables and TVs are projected to the floor and subtracted to give more accurate footprint estimates. SSCNet+ achieves higher *footprint* scores than SSCNet, but lower *freespace* scores.

5.3. Training from handheld camera footage

We additionally captured a 98,002-frame video dataset from an urban environment with a stereo camera. A model trained on this dataset allows us to make plausible predictions on images captured from a mobile phone camera on a different day (Figures 1 and 7).

5.4. Inference speed

Table 5 compares our inference speed with competing methods. For a fair comparison, all methods were assessed with a batch size of one. Our simple image-to-image architecture is significantly quicker than alternatives, lending itself more readily to mobile deployment.

6. Use case: Path Planning

One important use case for our system is to assist in the planning of paths, e.g. for an augmented reality character.

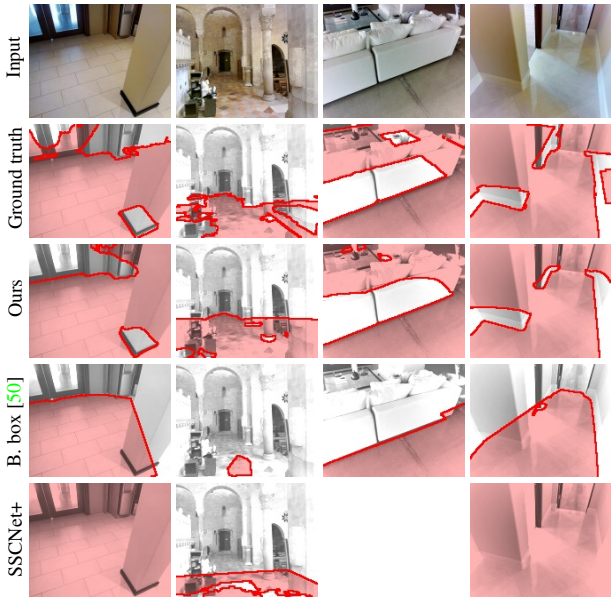


Figure 8. **Matterport results:** We predict the geometry of objects which do not fall into categories detectable by off-the-shelf object detectors, e.g. the pillar in the leftmost column. The rightmost column demonstrates how we can predict the continuation of traversable surfaces through doorways. No SSCNet+ results were computed for the third column due to layout estimation failure.

For each Matterport test image we choose a random pixel on the ground truth 'visible ground' mask as the start point and a pixel in the ground truth 'hidden ground' mask as the end point. We plan a path between the two with A* [26], where the cost of traversing pixel j is $1 - s_j^*$, where s_j^* is the unthresholded sigmoid output. A planned path is 'failed' if it leaves the ground truth traversable area at any point; we also count the fraction of pixels in each path which leave the ground truth traversable area as 'collisions'. Results are shown in Table 6, and examples of planned paths are shown in Figure 1 and in the supplementary material.

7. Conclusions

In this work we have presented a novel representation for predicting scene geometry beyond the line of sight, and we have shown how to learn to predict this geometry using only stereo or depth-camera video as input. We demonstrated our system's performance on a range of challenging datasets, and compared against several strong baselines. Future work could address temporal consistency or persistent predictions.

Acknowledgements Thanks to Eugene Valassakis for his help preparing this work's precursor [72]. Special thanks also to Galen Han and Daniyar Turmukhambetov for help capturing, calibrating and preprocessing our handheld camera footage, and to Kjell Bronder for facilitating the dataset annotation.

References

[1] S. Y. Bao, A. Furlan, L. Fei-Fei, and S. Savarese. Understanding the 3D layout of a cluttered room from multiple images. In *WACV*, 2014.

[2] H. Caesar, J. Uijlings, and V. Ferrari. COCO-stuff: Thing and stuff classes in context. In *CVPR*, 2018.

[3] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *3DV*, 2017.

[4] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In *CVPR*, 2018.

[5] J. W. S. Chong, S. Ong, A. Y. Nee, and K. Youcef-Youmi. Robot programming using augmented reality: An interactive method for planning collision-free paths. *Robot. Comp.-Integr. Manuf.*, 2009.

[6] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *ECCV*, 2016.

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[8] H. Dhamo, K. Tateno, I. Laina, N. Navab, and F. Tombari. Peeking behind objects: Layered depth prediction from a single image. *Pattern Recognition Letters*, 2018.

[9] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014.

[10] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow. Structured Prediction of Unobserved Voxels From a Single Depth Image. In *CVPR*, 2016.

[11] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981.

[12] P. Follmann, R. König, P. H. Rtinger, M. Klostermann, and T. Böttger. Learning to see the invisible: End-to-end trainable amodal instance segmentation. In *WACV*, 2019.

[13] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic. People watching: Human actions as a cue for single view geometry. *IJCV*, 2014.

[14] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018.

[15] R. Garg, V. Kumar BG, and I. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016.

[16] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012.

[17] J. J. Georgia Gkioxari, Jitendra Malik. Mesh R-CNN. In *ICCV*, 2019.

[18] G. Gerstweiler, K. Platzer, and H. Kaufmann. DARGS: dynamic AR guiding system for indoor environments. *Comp.*, 2018.

[19] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[20] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019.

[21] R. Guo and D. Hoiem. Beyond the line of sight: labeling the underlying surfaces. In *ECCV*, 2012.

[22] R. Guo and D. Hoiem. Support surface prediction in indoor scenes. In *ICCV*, 2013.

[23] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3D scene geometry to human workspace. In *CVPR*, 2011.

[24] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017.

[25] A. Harakeh, D. Asmar, and E. Shammas. Identifying good training data for self-supervised free space estimation. In *CVPR*, 2016.

[26] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 1968.

[27] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[28] V. Hedau, D. Hoiem, and D. Forsyth. Recovering free space of indoor scenes from a single image. In *CVPR*, 2012.

[29] J. Hu, M. Ozay, Y. Zhang, and T. Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In *WACV*, 2018.

[30] T.-W. Hui, X. Tang, and C. C. Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, 2018.

[31] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

[32] J. Ku, A. D. Pon, and S. L. Waslander. Monocular 3D object detection leveraging accurate proposals and shape reconstruction. In *CVPR*, 2019.

[33] J. J. Kuffner. Goal-directed navigation for animated characters using real-time path planning and control. In *Intell. Workshop Capt. Techn. Virt. Env.*, 1998.

[34] A. Kumar, S. Gupta, and J. Malik. Learning navigation subroutines by watching videos. *arXiv:1905.12612*, 2019.

[35] C.-Y. Lee, V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich. Roomnet: End-to-end room layout estimation. In *ICCV*, 2017.

[36] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NeurIPS*, 2010.

[37] P. Li, X. Chen, and S. Shen. Stereo R-CNN based 3d object detection for autonomous driving. In *CVPR*, 2019.

[38] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. T. Freeman. Learning the depths of moving people by watching frozen people. In *CVPR*, 2019.

[39] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3D object detection with RGBD cameras. In *ICCV*, 2013.

[40] C. Liu, P. Kohli, and Y. Furukawa. Layered scene decomposition via the occlusion-CRF. In *CVPR*, 2016.

[41] J. McCormac, A. Handa, A. Davison, and S. Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *ICRA*, 2017.

[42] A. Monszpart, P. Guerrero, D. Ceylan, E. Yumer, and N. J. Mitra. iMapper: Interaction-guided Joint Scene and Human Motion Mapping from Monocular Videos. *TOG*, 2018.

[43] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3D bounding box estimation using deep learning and geometry. In *CVPR*, 2017.

[44] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *Transactions on Robotics*, 2017.

[45] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *CVPR*, 2010.

[46] R. A. Newcombe, S. Izadi, and O. Hilliges. Kinectfusion: Real-time dense surface mapping and tracking. In *UIST*, 2011.

[47] S. Niklaus. A reimplementation of LiteFlowNet using PyTorch. https://github.com/sniklaus/pytorch-liteflownet, 2019.

[48] B. Pan, J. Sun, A. Andonian, A. Oliva, and B. Zhou. Cross-view semantic segmentation for sensing surroundings. *arXiv:1906.03560*, 2019.

[49] M. Poggi, F. Tosi, and S. Mattoccia. Learning monocular depth estimation with unsupervised trinocular assumptions. In *3DV*, 2018.

[50] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep Hough voting for 3D object detection in point clouds. *arXiv:1904.09664*, 2019.

[51] L. Qi, L. Jiang, S. Liu, X. Shen, and J. Jia. Amodal instance segmentation with KINS dataset. In *CVPR*, 2019.

[52] A. Ranjan, V. Jampani, K. Kim, D. Sun, J. Wulff, and M. J. Black. Adversarial collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. *arXiv:1805.09806*, 2018.

[53] T. Roddick, A. Kendall, and R. Cipolla. Orthographic feature transform for monocular 3D object detection. *BMVC*, 2019.

[54] A. M. Santana, K. R. Aires, R. M. Veras, and A. A. Medeiros. An approach for 2D visual occupancy grid map using monocular vision. *Theor. Com. Sci.*, 2011.

[55] S. Schulter, M. Zhai, N. Jacobs, and M. Chandraker. Learning to look around objects for top-view representations of outdoor scenes. In *ECCV*, 2018.

[56] L. A. Schwarz. Non-rigid registration using free-form deformations. *Technische Universität München*, 2007.

[57] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box in the box: Joint 3D layout and object reasoning from single images. In *ICCV*, 2013.

[58] R. Senanayake and F. Ramos. Building continuous occupancy maps with moving robots. In *AAAI*, 2018.

[59] T. Shao, A. Monszpart, Y. Zheng, B. Koo, W. Xu, K. Zhou, and N. J. Mitra. Imagining the unseen: Stability-based cuboid arrangements for scene understanding. *TOG*, 2014.

[60] D. Shin, Z. Ren, E. B. Sudderth, and C. C. Fowlkes. Multi-layer depth and epipolar feature transformers for 3D scene reconstruction. In *CVPR Workshops*, 2019.

[61] N. Silberman, L. Shapira, R. Gal, and P. Kohli. A contour completion model for augmenting surface reconstructions. In *ECCV*, 2014.

[62] S. Song and J. Xiao. Deep sliding shapes for amodal 3D object detection in RGB-D images. In *CVPR*, 2016.

[63] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017.

[64] S. Song, A. Zeng, A. X. Chang, M. Savva, S. Savarese, and T. Funkhouser. Im2Pano3D: Extrapolating 360° structure and semantics beyond the field of view. In *CVPR*, 2018.

[65] G. J. Stein, C. Bradley, and N. Roy. Learning over subgoals for efficient navigation of structured, unknown environments. In *Conference on Robot Learning*, 2018.

[66] A. Stentz. Optimal and efficient path planning for partially known environments. In *Intell. Unmnnd Grnd Veh.* 1997.

[67] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *IROS*, 2006.

[68] S. Tulsiani, R. Tucker, and N. Snavely. Layer-structured 3D scene inference via view synthesis. In *ECCV*, 2018.

[69] Y. Wang, P. Wang, Z. Yang, C. Luo, Y. Yang, and W. Xu. Unos: Unified unsupervised optical-flow and stereo-depth estimation by watching videos. In *CVPR*, 2019.

[70] Y. Wang, Y. Yang, Z. Yang, L. Zhao, P. Wang, and W. Xu. Occlusion aware unsupervised learning of optical flow. In *CVPR*, 2018.

[71] Z. Wang, B. Liu, S. Schulter, and M. Chandraker. A parametric top-view representation of complex road scenes. In *CVPR*, 2019.

[72] J. Watson. Inferring the footprints of objects in a single monocular image. Master's thesis, UCL, 2018.

[73] J. Watson, M. Firman, G. Brostow, and D. Turmukhambetov. Self-supervised monocular depth hints. In *ICCV*, 2019.

[74] G. Wayne, C. Hung, D. Amos, M. Mirza, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv:1803.10760*, 2018.

[75] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *ICRA workshops*, 2010.

[76] J. Xie, R. Girshick, and A. Farhadi. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. In *ECCV*, 2016.

[77] J. Yao, S. Ramalingam, Y. Taguchi, Y. Miki, and R. Urtasun. Estimating drivable collision-free space from monocular video. In *WACV*, 2015.

[78] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ADE20K dataset. *CVPR*, 2017.

[79] T. Zhou, M. Brown, N. Snavely, and D. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

[80] Y. Zhu, Y. Tian, D. Metaxas, and P. Dollár. Semantic amodal segmentation. In *CVPR*, 2017.

[81] Y. Zou, Z. Luo, and J.-B. Huang. DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In *ECCV*, 2018.